NLPX - An XML-IR System with a Natural Language Interface

Alan Woodley

Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
Queensland 4001 Australia

ap.woodley@student.qut.edu.au

Shlomo Geva

Centre for Information Technology Innovation
Faculty of Information Technology
Queensland University of Technology
Queensland 4001 Australia

s.geva@qut.edu.au

Abstract *Traditional information retrieval (IR) systems respond to user queries with ranked lists of relevant documents. The separation of content and structure in XML documents allows individual XML elements to be selected in isolation. Thus, users expect XML-IR systems to return highly relevant results that are more precise than entire documents. This paper presents such a system. The system accepts queries in both natural language (English) and formal XPath-like format (NEXI) and matches to a set of relevant and appropriately-sized elements using an effective ranking scheme.*

Keywords Information Retrieval, Natural Language Queries

1.0 Introduction

The widespread use of Extensible Markup Language (XML) documents in digital libraries has lead to development of information retrieval (IR) methods specifically designed for XML collections. Most traditional IR systems are limited to whole document retrieval; however, since XML documents separate content and structure, XML-IR systems are able to retrieve the relevant portions of documents. Users interacting with XML-IR system could potentially receive highly relevant and highly precise material. However, it also means that XML-IR systems are more complex than their traditional counterparts.

We describe a system that attempts to solve some of the challenging problems of XML-IR. In what follows, we first describe how queries are interpreted by the system. Two query formats are examined: natural language, and NEXI queries, an XPath variant where users express their information need in a formal language. We then very briefly describe the internal storage structure of the XML collection and the ranking scheme that is used to order results. Finally we present some performance results from the INEX 2004 Workshop.

Proceedings of the 9th Australian Document Computing Symposium, Melbourne, Australia, December 13, 2004.

2.0 Query Interpretation

The system presented here was designed to participate in the 2004 Initiative for the Evaluation of XML Retrieval (INEX) Workshop [2]. The INEX Workshop is similar to the TREC workshop. It is an annual event that provides a world-class benchmark for the evaluation of XML systems. INEX provides a test collection of 12,000 IEEE journal articles, a set of queries and a set of evaluation metrics. Two types of queries are used in INEX: CO and CAS. Content Only (CO) queries ignore document structure and only contain content requirements. Contrastly, Content and Structure (CAS) queries explicitly express both content and structural requirements. Both CO and CAS queries are expected to return appropriately sized elements – not just whole documents. Figures 1 and 2 are examples of both query types.

```
<inex_topic topic_id="XX" query_type="CO">
<title>
  "multi layer perceptron" "radial basis
functions" comparison
</title>
<description>
  The relationship and comparisons between
radial basis functions and multi layer
perceptrons
</description>
</inex_topic>
```

Figure 1 A CO Query

Both the description and title tags express users' information needs. The description expresses users' need in a natural language (e.g. English). The title expresses users' information need in either a list of keywords/phrases (CO) or as a formal XPath-like language (CAS) called Narrowed Extended XPath I (NEXI) [5].

```
<inex_topic topic_id="XX"
query_type="CAS">
<title>
  //article[about(.,information
retrieval)]//sec[about(.,compression)]
</title>
<description>
  Find sections about compression in
articles about information retrieval.
</description>
</inex_topic>
```

Figure 2 A CAS Query

NEXI's syntax is **//A[about(//B,C)]** where **A** is the context path, **B** is the relative path and **C** is the content requirement. Each **'about'** clause represents an individual information request. So the query **//A[about(//B,C)]//X[about(//Y,Z)]** contains two requests: **//A[about(//B,C)]** and **//A//X[about(//Y,Z)]**. However, in NEXI only elements matching the leaf (i.e. rightmost) **'about'** clause are returned to the user, and the others are used to support the return elements in ranking.

In 2004 INEX introduced its natural language track. At the INEX 2003 Workshop more than two-thirds of the proposed queries had major semantic or syntactic errors [4] that required 12 rounds of corrections. Since experts in the field of structured information retrieval are unable to easily use formal query languages, one cannot expect an inexperienced user to do so. However, most users are able to intuitively express their information need in a natural language. There already exists an extensive body of research into natural language processing in the specific area of Information Retrieval, largely thanks to The Text Retrieval Conference (TREC) and the Special Interest Group for Information Retrieval (ACM-SIGIR). However, work on an XML-IR interface is still largely un-documented and many problems remain unsolved.

2.1 Natural Language Query (NLQ) to NEXI Translator

Our system was originally developed for participation in the Ad-hoc track using NEXI. We adapted it to handle natural language queries by converting NLQs to NEXI.

Step 1 Lexical and Semantic Tagging

Suppose that the description tags in Figure 1 and 2 are input into the system as natural language queries (NLQ). Translating the NLQs into NEXI format takes several steps. First each word is tagged as either as a special connotation or by its part of speech. Special connotations are words of implied semantic significance within the system. Our system uses three types of special connotations: structural words that indicate the structural requirement of the user (e.g. article, section, paragraph, etc.), boundary words that separate the user's structural and content requirements (e.g. about, containing) and instruction words that indicate if we have a return or support request. All other words are tagged by their part of speech. Any part-of-speech tagger could perform this task; however, our system uses the Brill Tagger [1]. Figure 3 is an example of the NLQ after tagging.

NLQ 1: The/DT relationship/NN and/CC comparisons/NNS between/IN radial/JJ basis/NN functions/NNS and/CC multi/NNS layer/NN perceptions/NN
NLQ 2: Find/XIN sections/XST about/XBD compression/NN in/IN articles/XST about/XBD information/NN retrieval/NN

Figure 3 A Tagged CO and CAS Natural Language Query

Step 2 Template Matching

The translator's second task is to derive information requests from the tagged NLQ by matching the tagged NLQ to a predefined set of grammar templates. The grammar templates were developed by inspection of previous years' INEX queries. NLQs have a narrow context and require the understanding of only a subset of natural language. A system that interprets NLQs requires fewer rules than a system that attempts to understand natural language in general. Inspection of previous INEX queries reveals that most queries correspond to a small set of patterns. By extracting these patterns we were able to formulate grammar templates that matched a majority of queries. Figure 4 shows some of the grammar templates.

Query: Request+
Request : CO_Request | CAS_Request
CO_Request: NounPhrase+
CAS_Request: SupportRequest | ReturnRequest
SupportRequest: Structure [Bound] NounPhrase+
ReturnRequest: Instruction Structure [Bound] NounPhrase+

Figure 4 Grammar Templates

Each grammar template corresponds to an individual information request. Each

information request has three attributes: **Content,** a list of terms or phrases expressing users content requirements, **Structure,** a logical XPath expression that describes the structural constraints of the request. And **Instruction,** "R" if we have a return request or "S" if we have a support request. Figure 5 is an example of the information requests derived from the templates.

NLQ 1:
Structure: /*
Content: relationship, comparisons, radial basis functions, multi layer perceptions
Instruction: R

NLQ 2:

	Request 1	**Request 2**
Structural:	/article/sec	/articlec
Content:	compression	information retrieval
Instruction:	R	S

Figure 5 Derived Information Requests

Step 3 NEXI Query Production

The final step in the translator is to merge the information request into a single NEXI query. Return requests are output in the form **A[about(.,C)]** where A is the request structural attribute and C is the request content attribute. To add support requests, we must first locate the longest matching string in the return request and then add the support request in the form **D[about(E,F)]** where D is the longest matching string, E is the remainder of the support request structural attribute and F, is the support requests content attribute.

Figure 6 is how the NEXI queries would appear after the information requests for each NLQ have been merged.

NLQ 1:
//*[about(**.,**relationship, comparisons, radial basis functions, multi layer perceptions)]
NLQ 2:
//article[about(.,information retrieval)]//sec[about(.,compression)]

Figure 6 NLQ-to-NEXI Queries

2.2 Processing NEXI Queries

Once NEXI queries are input into the system they are converted into an intermediate language called the RS query language. The RS query language converts NEXI queries to a set of information requests. The format of RS queries is

Request: Instruction 'l' Retrieve_Filter 'l' Search_Filter 'l' Content.

The Instruction and Content attributes are the same as they were in the previous section; however, the Structural attribute has been divided into a Retrieve and Search Filter. While both are logical XPath expressions the **Retrieve Filter** describes which elements should be retrieved by the system, while, the **Search Filter** describes which elements should be searched by the system. Figure 7 is an example of the queries introduced earlier converted to RS queries.

RS Query 1:
R|//*|//*| relationship, comparisons, radial basis functions, multi layer perceptions
RS Query 2:
R|//article//sec|//article//sec|compression
S|//article|//article| information retrieval

Figure 7 An Example of an RS Query

3.0 System Structure

We index the XML collection using an inverted list. Given a query term we can derive the filename, physical XPath and the ordinal position within the XPath that it occurred in. From there we construct a partial XML tree containing every relevant leaf element for each document that contains a query term. Further information on our structure can be found in [3].

4.0 Ranking Scheme

Elements are ranked according to their relevance. Data in an XML tree is mostly stored in leaf elements. So first we calculate the score of relevant leaf elements, then, we propagate their scores to their ancestor branch elements.

The relevance score of leaf elements is computed from term frequencies within the leaf elements normalised by their global collection frequency. The scoring scheme rewards elements with more query terms. However, it penalises elements with frequently occurring query terms, and rewards elements that contain more distinct query terms.

The relevance score of a non-leaf is the sum of the children scores. However leaf element scores are moderated by a slight decay

factor as they propagate up the tree. Branch elements with multiple relevant children are likely to be ranked higher then their descendents – as they are more comprehensive - while branch elements with a single relevant child will be ranked lower than the child element as they are less specific.

5.0 Results

The system was entered into both the Ad-hoc and NLP tracks at INEX2004. In the Ad-hoc track the system ranked 1st from 52 submitted runs in the VCAS task, and 6th from 70 submitted runs in the CO task. In the NLP track the system was ranked 1st in the VCAS task and 2nd in the CO task. While the NLP track was limited to 9 participants initially, of which only 4 made official submissions, the most encouraging outcome was that the NLP system outperformed several Ad-Hoc systems. In fact, if the NLP submission was entered in the Ad-hoc track it would have ranked 12th from 52 in VCAS and 13th from 70 in CO. This seems to suggest that in structured IR, natural language queries have the potential to be a viable alternative, albeit not as precise, to a formal query language such as NEXI (an XPath derivative).

The Recall/Precision Curves for the Ad-hoc track, along with the R/P curve for our NLP runs are presented in Figures 8 and 9. The top bold curve is the Ad-hoc curve, the lower is the NLP curve, and the background curves are of all the official Ad-hoc runs at INEX 2004.

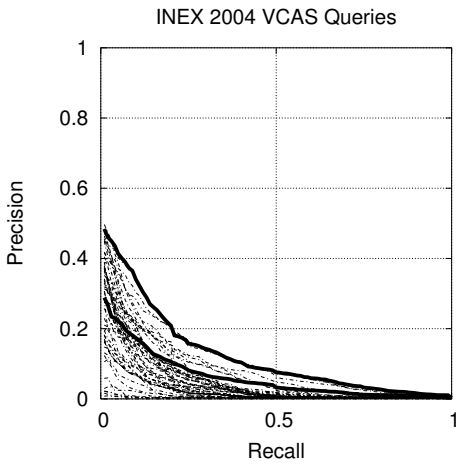

Figure 8 The INEX 2004 VCAS R/P Curve

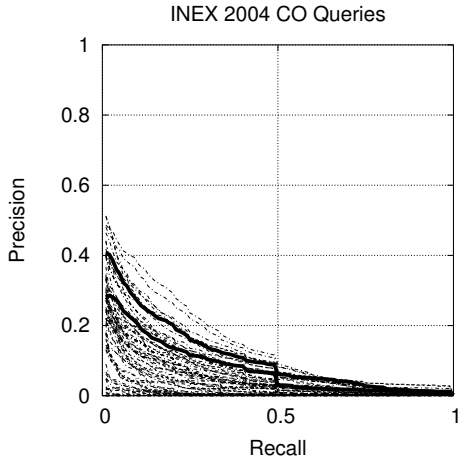

Figure 9 The 2004 INEX CO R/P Curve.

6.0 Conclusion and Future Outlook

This paper presents an XML-IR system responds to user queries with relevant and appropriately sized results. Our ranking scheme is comparable with the best INEX alternatives. The NLP interface requires further development; however, initial results are promising. The system provides a working example of the potential of XML-IR systems.

References

[1] E. Brill. A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the Third Conference on Applied Computational Linguistics* (ACL), Trento, Italy. 1992.

[2] N. Fuhr and S. Malik. Overview of the Initiative for the Evaluation of XML Retrieval (INEX) 2003. In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17, 2003, pages 1-11. 2004.

[3] S. Geva and M. Spork. XPath Inverted File for Information Retrieval, In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17,2003, pages 110-117. 2004.

[4] Trotman, A. and O'Keefe, "The Simplest Query Language That Could Possibly Work", In *INEX 2003 Workshop Proceedings*, Schloss Dagstuhl, Germany, December 15-17,2003, pages 167-174, 2004.

[5] A. Trotman and B. Sigurbjörnsson, *Narrowed Extended XPath I (NEXI)*, http://www.cs.otago.ac.nz/postgrads/andrew/2004-4.pdf, 2004.